Analytical Characterization of the Throughput of a Split TCP Connection

Ananth I. Sundararaj and Dan Duchamp

Department of Computer Science
Stevens Institute of Technology
Hoboken, NJ 07030 USA
{asundara,djd}@cs.stevens-tech.edu

***Abstract*— We introduce the idea of implementing a single logical end-to-end connection as a series of cascaded transport connections and argue that there are several potential advantages to doing so. One possible advantage investigated in this paper is improved throughput via pipeline parallelism. We develop an equation for the throughput of such a *split connection* as a function of packet loss rate, average round trip time, and the receiver's advertised window. Our work builds upon the model developed by Padhye et al. [1]. We extend that work by fixing a small error and modeling two TCP connections in series, making the important extension of including slow start in the analysis. We validated our analysis by comparing results against ns-2 simulations. We studied throughput over wide ranges of packet loss and round trip time. We found that, for loss probabilities of up to 0.005, a split TCP connection has up to one and a half times greater throughput than a single TCP connection.**

I. Introduction

The Internet is increasingly being populated by intermediaries. For example, located between a browser and a web server there might be a firewall on the client side, one or more web caches at points in the network, and a workload balancer on the server side. TCP provides transport for HTTP, and it is the protocol that is responsible for verifying end-to-end delivery. Therefore, each intermediary must inspect and possibly change payload and/or TCP headers, and must do so transparently. Such a method of operation, in addition to being inelegant, makes intermediate services harder to write, thereby constraining the services that developers build or even contemplate.

Accordingly, we are engaged in research that seeks to make it easier to write and deploy network intermediaries by providing a cleaner programming model. In particular, we are examining the advantages and disadvantages of relieving transport protocols of the burden of end-to-end delivery verification. In this new architecture, a higher level protocol located between the transport and application layers would verify end-to-end delivery according to the semantics that the application desires, and would explicitly identify which application or protocol fields should be visible to which intermediaries. This new end-to-end protocol would run on top of one or more transport connections, which would be connected in series. If intermediaries were present between endpoints, one possible mode of operation would be to use separate transport connections between each pair of nodes; for example, if X and Y were the endpoints, and Z were an intermediary, then two transport connections, X-Z and Z-Y, would be used in this mode of operation. The end-to-end protocol would ensure delivery between X and Y, and intermediate node Z would benefit from a simpler programming model of two terminated TCP connections.

Besides providing a cleaner programming model for intermediaries, dividing an end-to-end conversation across several transport connections promises many potential advantages, complete discussion of which is beyond the scope of this paper. The one potential advantage that we examine in this paper is the possibility of improved TCP throughput via pipeline parallelism: if data flows from X to Y through Z, then two TCP sender-receiver pairs operate over separate paths, filling two receive buffers. With proper timing, it is possible that the two transport connections could work in parallel for long enough to deliver data to Y faster than would a single transport connection between X and Y. Indeed, such a result has been reported, discovered by accident [9].

To investigate this possibility, we extended a well known analytic model of TCP [1] to obtain a closed form expression for the throughput of two TCP connections in series as a function of the packet loss probability, the average round trip time (RTT), and the receiver's advertised window size. The contributions of this work include:

1) Correcting a small error in the modeling of timeouts in Padhye et al. [1].
2) Slightly generalizing the model so that the average number of packets sent between two timeouts in sequence for the same unacknowledged packet is a parameter, *d*. Padhye et al. assume *d=1*.
3) Significantly extending the model to include slow start periods. Slow start is important for the transfer of small files (as in HTTP) and as packet losses increase. For

small files, the entire transfer may occur within the slow start phase. As losses increase, timeouts increase; in TCP, timeouts re-initiate slow start. These types of connections spend a considerable amount of time in the slow start stage, thus making it important to model.

4) Using this corrected and extended model to develop a closed form expression for the throughput of two TCP connections in series as a function of the packet loss rate, average RTT, and receiver's advertised window size.
5) Comparing our results to those obtained via simulations, and explaining the underlying reasons for discrepancies.

We found that, for common loss probabilities (up to 0.005), a split TCP connection has a throughput of up to one and half times that of a single TCP connection.

II. Previous Work

There are two sorts of related prior work: TCP modeling, and proposals for how to "split a TCP connection" as we have done, using two separate connections in series. There is no prior work on both; i.e., ours is the first analytical characterization of the throughput of split (or "cascaded") TCP connections.

TCP modeling has become popular in the last decade. One line of work has focused on modeling the stationary behavior of TCP in congestion avoidance mode. Ott, Mathis, and others developed the first model [14,10], which analyzed TCP's "fast retransmit" behavior wherein three consecutive duplicate ACKs trigger retransmission without closing the congestion window. Padhye et al. [1] extended the model to include the possibility of timeout (which does close the congestion window, thereby complicating the analysis). We extend the model to encompass slow start as well.

Altman, Avrachenkov, and Barakat have considered more general models [2,12] in which loss events may follow general distributions, allowing for correlations, burstiness, etc. Their results agree with those mentioned above in cases where the assumptions about loss events agree.

Previous work on split TCP connections was motivated by the desire to cope with connections running over vastly different communication media. The idea is to split the connection at the boundary of separate communication media so as to isolate issues pertaining to the different media. The original work in this vein is I-TCP [3,4]. The problem addressed by I-TCP is that a TCP connection may experience needless closings of the congestion control window if one endpoint is connected to a fast network while the other endpoint is mobile and connected to slow wireless links. (Similar arguments apply to networks that include satellite links [5].) The mobile endpoint can cause the fixed endpoint's window to close in two ways. One is via mobility – moving from one link to another is typically accompanied by "handoff delay." During handoff, in-flight TCP segments are likely to be lost, leading to resumption of slow start. The second problem is that many wireless links have bit error rates – and hence packet loss rates – orders of magnitude higher than wired networks. I-TCP used what was effectively two cascaded TCP connections in series, one running between one endpoint and the wireless base station, and the second running over the wireless link between the base station and the mobile endpoint. The connection running on the wireless link was parameterized to cope better with the inherent high delay and loss of the wireless medium.

The problem with the I-TCP proposal – unlike in our work – was that there was no protocol above TCP to verify end-to-end delivery. Consequently, the semantics of two cascaded TCP connections were not the same as those of a single end-to-end connection: when the fixed endpoint received an ACK, it signaled delivery to the base station, not necessarily to the mobile host. A proposed solution is TCP spoofing [6]. In TCP spoofing, the base station spies on the TCP connection. It resends packets lost on the wireless link, and sends back to the fixed endpoint manufactured ACKs in order to give the source the illusion of a lossless, short delay path. TCP spoofing has been found to improve throughput, but suffers from the requirement that ACKs flow through the same path as the data. (I-TCP also delivers better TCP throughput, but it addresses difficulties that are unique to the wireless environment. Our work applies to a general environment.)

Extensive discussion of the use of proxies to mitigate link-specific effects can be found in a recent RFC on the subject [13]. That report contains many more examples, further citations of related work, and a survey and taxonomy of "performance-enhancing proxies" (PEPs).

Recently, there has been a great deal of interest in the use of transparent proxies at the application level. Prominent examples are Web caches and "transcoders" of content for display on handheld devices. Transparent proxies are commonly used when an application is to be proxied in a manner that is completely oblivious to a client, without requiring any prior configuration. To ensure that all IP packets of an intercepted TCP connection are seen by the intercepting transparent proxy, the proxy must sit at focal points in the network. Work to overcome this limitation using TCP options and IP tunneling was, in part, the inspiration for our own work; that study [9] found, by accident, that a split TCP connection performed better than a single connection between the same endpoints.

III. Background

Padhye et al. developed the following expression for TCP throughput

$$B(p) = \begin{cases} \dfrac{\frac{1-p}{p} + E[W_u] + Q\prime(E[W_u])\frac{1}{1-p}}{RTT\left(\frac{b}{2}E[W_u]+1\right) + Q\prime(E[W_u])T_o\frac{f(p)}{1-p}} & if\ E[W_u] < W_{max} \\ \dfrac{\frac{1-p}{p} + W_{max} + Q\prime(W_{max})\frac{1}{1-p}}{RTT\left(\frac{b}{8}W_{max} + \frac{1-p}{pW_{max}} + 2\right) + Q\prime(W_{max})T_o\frac{f(p)}{1-p}} & otherwise \end{cases}$$

where: $B(p)$ is the TCP throughput;

p is the probability that a packet is lost;

$f(p)$ is defined as

$f(p) = 1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6$;

$E[W_u]$ is the expected value of the unconstrained congestion control window size;

W_{max} is the maximum receiver advertised window;

$Q\prime(w)$ is the probability that a loss in a window of size *w* is a timeout;

T_o is the time that a sender waits before retransmitting; and

b is the number of packets acknowledged by each ACK.

Padhye et al. studied TCP Reno, and they assumed that the time spent in slow start is negligible compared to the length of a connection, that the packet losses in a round are correlated, and that RTT is independent of the window size.

Their model describes TCP's congestion avoidance phase, wherein the congestion control window is increased each time an ACK is received. Conversely, the window is decreased whenever a lost packet is detected, with the amount of the decrease depending on whether packet loss is detected by duplicate ACKs or by timeout. TCP's congestion avoidance behavior is modeled in terms of *rounds*. A round is, roughly, the transmission of a window of packets ending with either acknowledgement or indication of loss; further definition is given in Section VI. A stochastic model of TCP congestion control is developed in several steps: first when loss indications are exclusively triple duplicate ACKs, then when loss indications may be either triple duplicate ACKs or timeouts, and finally when the congestion window size is limited by the receiver's advertised window. Most of the notation and analysis used in our work is the same as they developed; however, for the sake of completeness, and in order to introduce our changes, we develop our extended model from first principles. Windows are sized in terms of packets, not bytes, and when we use the word throughput, we do not refer to the good-put of the connection. We are interested in the total packets sent in the connection irrespective of their eventual fate.

The remainder of the paper proceeds as follows. Section IV corrects a small error made by Padhye et al.; this is our first contribution. Sections V through VII model the throughput of two consecutive cascaded connections, incorporating the correction and our second contribution, a small generalization of one aspect of the model. Sections VIII and IX introduce our third contribution, an analysis of the slow start phase. Section X succinctly restates our closed form expression for "split connection" throughput – our fourth contribution. Section XI compares analytical and simulation results, our final contribution. Section XII concludes.

IV. Error in [1]

In [1], $Q\prime(w)$, the probability that a loss in a window of size *w* is a timeout, is described as

$$Q\prime(w) = \begin{cases} 1 & w \leq 3 \\ \sum_{k=0}^{2} A(w,k) + \\ \left(\sum_{k=3}^{w} A(w,k) * \sum_{m=0}^{2} C(k,m)\right) & w > 3 \end{cases}$$

where $A(w,k)$ denotes the probability that the first *k* packets are acknowledged in the penultimate round of *w* packets, and $C(n,m)$ is the probability that *m* packets are acknowledged in sequence in the last round (where *n* packets are sent). But when $w > 3$, $Q\prime(w)$ should be defined as $Q\prime(w) = \sum_{k=0}^{2} A(w,k) + \left(\sum_{k=3}^{w-1} A(w,k) * \sum_{m=0}^{2} C(k,m)\right)$. That is, the second summation, $\sum_{k=3}^{w} A(w,k)$, in the expression should be $\sum_{k=3}^{w-1} A(w,k)$. Otherwise, when $k = w$, we would have $A(w,w)$; i.e., all *w* packets sent in the penultimate round are acknowledged. However, the round would be penultimate if and only if a loss occurs during that round. Hence the summation must be $A(w,3) + A(w,4) + ... + A(w,w-1)$ or $\sum_{k=3}^{w-1} A(w,k)$.

V. Modeling Throughput of a Split TCP Connection

We reduce the modeling of two cascaded connections to the modeling of only the second connection, according to the following argument. The throughput during an observed interval is the number of packets sent during that interval divided by the length of the interval. When the interval ends, each packet that was sent during the interval will either be lost, in flight over the first connection, in process at the intermediary, in flight over the second connection, or delivered to the endpoint. Ignoring the packets in flight over the first connection and in process at the intermediary considerably simplifies the analysis and is justified by the likelihood that the numbers of such packets will be small in practice.

Let Z_i^{SS} denote the duration when the connection is in slow start, Z_i^{TO} denote the duration of a sequence of timeouts and Z_i^{TD} denote the duration of a sequence of triple duplicate periods, the last of which ends in a timeout. A Triple Duplicate Period (TDP) is defined as the period between two Triple Duplicate (TD) loss indications, or between a TD loss indication and a timeout loss indication without any other loss indications in between.

Define

$$S_i = Z_i^{SS} + Z_i^{TO} + Z_i^{TD}$$

and M_i to be the number of packets sent during S_i.

If $(S_i, M_i)_i$is an independent identically distributed sequence of random variables [8], then the throughput of the connection, B, is defined as

$$B = \frac{E[M]}{E[S]} \tag{1}$$

Let n_i be the number of TD periods in Z_i^{TD}. Then for the j^{th} TD period of the interval Z_i^{TD}, we define Y_{ij} to be the number of packets sent in the period, A_{ij} to be the duration of the period, X_{ij} to be the number of rounds in the period, and W_{ij} to be the window size at the end of the period.

Further, let R_i denote the total number of packets transmitted in Z_i^{TO} and L_i the number of packets sent in Z_i^{SS}. Then we have

$$M_i = \sum_{j=1}^{n_i} Y_{ij} + R_i + L_i$$

$$S_i = \sum_{j=1}^{n_i} A_{ij} + Z_i^{TO} + Z_i^{SS}$$

Now, if $\{n_i\}_i$ is an independent identically distributed sequence of random variables, independent of $\{Y_{ij}\}$ and $\{A_{ij}\}$ then we have

$$E[M] = E[n]E[Y] + E[R] + E[L] \tag{2}$$

and

$$E[S] = E[n]E[A] + E\left[Z^{TO}\right] + E\left[Z^{SS}\right] \tag{3}$$

To arrive at $E[n]$, we note that in Z_i^{TD} there are n TDPs, and the first $(n-1)$ TDPs end in a TD while the last TDP ends in a timeout. So in a Z_i^{TD}, there is one timeout loss indication out of n_i loss indications. Let Q represent the probability that the loss indication ending a TDP is a timeout loss indication, so

$$Q = \frac{1}{E[n]} \tag{4}$$

Using (1), (2), (3), and (4) we have

$$B = \frac{E[Y] + Q * (E[R] + E[L])}{E[A] + Q * (E\left[Z^{TO}\right] + E\left[Z^{SS}\right])} \tag{5}$$

VI. Congestion Avoidance and Fast Retransmit

In this section we will obtain expressions for $E[Y]$ and $E[A]$. Let *W* denote TCP's congestion control window size. The congestion avoidance mechanism is modeled in terms of rounds. A round starts with the back to back transmission of *W* packets. Once all the packets falling within the congestion control window have been sent, no other packet is sent until the first ACK is received for one of these *W* packets. The receipt of this ACK marks the end of the round and the beginning of the next round. The duration of the round is the same as the round trip time and is assumed to be independent of the window size. Further, it is assumed that the time required to send *W* packets is smaller than the round trip time. The assumptions made are that a packet is lost in a round independently of any other packet lost in any other round and that if a packet is lost then all remaining packets transmitted until the end of the round are lost as well. We define p to be the probability that a packet is lost, given that it is the first packet in its round or that the preceding packet in its round is not lost.

A. Derivation of E[Y]

A TD period starts immediately after a TD loss indication. Considering the i^{th} TD period: the current congestion control window is $\frac{W_{i-1}}{2}$, α_i denotes the first packet lost in the i^{th} TDP – i.e. until such time and including it, α_i packets have been transmitted – and X_i denotes the round where this loss occurred. After packet α_i, a maximum of $W_i - 1$ more packets might be sent in an additional round before the current TD period ends. Thus a maximum of $Y_i = \alpha_i + W_i - 1$ packets are sent in $X_i + 1$ rounds. It follows that

$$E[Y] = E[\alpha] + E[W] - 1 \tag{6}$$

Consider the random process $\{\alpha_i\}_i$, which is a sequence of independent and identically distributed random variables, since the packets are lost independently of any other packets lost in any other rounds. The probability that $\alpha_i = k$ is equal to the probability that exactly $(k-1)$ packets are successfully acknowledged before a loss occurs, hence

$$P[\alpha = k] = (1-p)^{2(k-1)}\left(1 - (1-p)^2\right),\ k = 1, 2, 3\ldots$$

The mean of α is

$$E[\alpha] = \sum_{k=1}^{\infty} (1-p)^{2(k-1)}\left(1 - (1-p)^2\right) k$$

or

$$E[\alpha] = \frac{1}{\left(1 - (1-p)^2\right)} \tag{7}$$

At the beginning of the last round the window size would be W_i, where

$$W_i = \frac{W_{i-1}}{2} + \frac{X_i}{b},\ i = 1, 2, 3\ldots \tag{8}$$

Note the assumption that $\frac{W_{i-1}}{2}$ and $\frac{X_i}{b}$ are integers. So the total number of packets sent would be

$$Y_i = \frac{X_i}{2} * \left[W_i + \frac{W_{i-1}}{2} - 1\right] + \beta_i \tag{9}$$

where β_i is the number of packets sent in the last round. Next, if we assume $\{X_i\}$ and $\{W_i\}$ to be mutually independent sequences of independent identically distributed random variables, we have

$$E[W] = \frac{2}{b} * E[X] \tag{10}$$

As β_i can be assumed to be uniformly distributed between 0 and W_i , we have

$$E[Y] = \frac{b}{4}E[W]\left(\frac{E[W]}{2} + E[W] - 1\right) + \frac{E[W]}{2} \quad (11)$$

Substituting (6) and (7) in (11) and solving for $E[W]$ we have

$$E[W] = \frac{(2+b)}{3b} + \sqrt{\left(\frac{b+2}{3b}\right)^2 + \frac{8}{3b}\left(\frac{(1-p)}{(1-(1-p)^2)}\right)} \quad (12)$$

Substituting (7) and (12) in (6) we have

$$E[Y] = \frac{(1-p)^2}{\left(1-(1-p)^2\right)} + \frac{(2+b)}{3b} + \sqrt{\left(\frac{b+2}{3b}\right)^2 + \frac{8}{3b}\left(\frac{(1-p)}{\left(1-(1-p)^2\right)}\right)} \quad (13)$$

From (10) and (12) we have

$$E[X] = \frac{(2+b)}{6} + \sqrt{\left(\frac{b+2}{6}\right)^2 + \frac{2b}{3}\left(\frac{(1-p)}{\left(1-(1-p)^2\right)}\right)} \quad (14)$$

B. Derivation of E[A]

Consider the i^{th} TDP, and let r_{ij} be the duration of the j^{th} round of TDP_i . Hence the duration of TDP_i is

$$A_i = \sum_{j=1}^{X_i+1} r_{ij}$$

Since r_{ij} are considered to be random variables, assumed to be independent of the size of the congestion window and hence independent of the round number *j*, it follows that

$$E[A] = (E[X]+1)\,E[r] \quad (15)$$

$E[r]$ is denoted by RTT, the average value of the round trip time for the connections in series under consideration, which in our case is the second connection of the split TCP connection. Hence substituting (14) in (15) we have

$$E[A] = RTT\left(\frac{(2+b)}{6} + \sqrt{\left(\frac{b+2}{6}\right)^2 + \frac{2b}{3}\left(\frac{(1-p)}{\left(1-(1-p)^2\right)}\right)} + 1\right) \quad (16)$$

VII. Timeout and Retransmission

In this section we will obtain expressions for $E[R]$ and $E[Z^{TO}]$. *Q* is the probability that a loss indication is a TO. We define by $Q\prime(w)$, the probability that a loss indication in a window size *w* is a time out. If $A(w,k)$ denotes the probability that the first *k* packets of *w* packets are ACKed in the penultimate round, then

$$A(w,k) = (1-p)^{2k}\left(1-(1-p)^2\right)$$

Also $C(n,m)$ is the probability that *m* packets are ACKed in sequence in the last round, where *n* packets are sent, then

$$C(n,m) = (1-p)^{2m}\left(1-(1-p)^2\right)$$

If the window size in the penultimate round is less than or equal to three, then three duplicate acknowledgments can never be received. Hence the loss indication will always be a timeout. But if the window size in the penultimate round is greater than 3, then a timeout will occur in either of the following two cases: (1) only two or fewer packets sent in the penultimate round are acknowledged, hence only another two packets may be sent in the last round and three duplicate ACKs can never be received, or (2) three or more packets sent in the penultimate round are acknowledged, but only two or fewer packets in the last round are successfully sent – then too three duplicate acknowledgments can never be received.

So

$$Q\prime(w) = \begin{cases} 1 & w \le 3 \\ \sum_{k=0}^{2} A(w,k) + \\ \left(\sum_{k=3}^{w-1} A(w,k) * \sum_{m=0}^{2} C(k,m)\right) & w > 3 \end{cases}$$

or

$$Q\prime(w) = \min\left(1,\ \left(1-(1-p)^6\right) * \left[1+(1-p)^6\left(1-(1-p)^{(2w-6)}\right)\right]\right) \quad (17)$$

Q, the probability that a loss indication is a time out, is given by

$$Q = \sum_{w=1}^{\infty} Q\prime(w) P[W=w]$$

or we write

$$Q = E\left[Q\prime\right]$$

or approximating it we have

$$Q = Q\prime\left(E[W]\right) \tag{18}$$

A. Derivation of $E[R]$

A maximum number of $(1+2+3) = 6$ packets are transmitted between two timeouts in sequence for the same unacknowledged packets. Let its average be denoted by a constant *d*. So if a sequence of *k* timeouts occur, then there are $(k-1)d$ consecutive losses, followed by a successfully transmitted packet.

So

$$P[R=k] = \left(1-(1-p)^2\right)^{(k-1)d} * (1-p)^2$$

$$E[R] = \sum_{k=1}^{\infty} kP[R=k]$$

or

$$E[R] = \frac{(1-p)^2}{\left(1-\left(1-(1-p)^2\right)^d\right)^2} \tag{19}$$

B. Derivation of $E\left[Z^{TO}\right]$

We know that in any sequence of timeouts, the first six have durations T_o , $2T_o$, $4T_o$, $8T_o$, $16T_o$, $32T_o$ and all following timeouts have duration $64T_o$. A sequence of *k* timeouts will have a duration

$$L_k = \begin{cases} \left(2^k-1\right)T_o & k \leq 6 \\ \left(63+64\left(k-6\right)\right)T_o & k \geq 7 \end{cases}$$

We have

$$E\left[Z^{TO}\right] = \sum_{k=1}^{6} L_k P[R=k] + \sum_{k=7}^{\infty} L_k P[R=k] \tag{20}$$

where

$$\sum_{k=1}^{6} L_k P[R=k] = (1-p)^2 T_o *$$
$$\sum_{k=1}^{6}\left(2^k-1\right) * \left(1-(1-p)^2\right)^{(k-1)d}$$

and

$$\sum_{k=7}^{\infty} L_k P[R=k] = (1-p)^2\left(1-(1-p)^2\right)^{6d} *$$
$$T_o\left[\frac{127-63\left(1-(1-p)^2\right)^d}{\left(1-\left(1-(1-p)^2\right)^d\right)^2}\right]$$

VIII. Slow Start Behavior of TCP

A. Derivation of E[L]

Let *W* denote TCP's congestion control window size. TCP's slow start behavior is modeled in terms of rounds as detailed in section VI. We denote by α_i the first packet lost in the slow start stage, i.e. until such time and including it, α_i packets have been transmitted, and X_i denotes the round where this loss occurs. After packet α_i, a maximum of $W_i - 2$ more packets might be sent in an additional round before the current slow start stage comes to an end. Thus a maximum of $L_i = \alpha_i + W_i - 2$ packets are sent in $X_i + 1$ rounds. We note that the slow start might also come to an end when the *cwnd* variable exceeds the *ssthresh* variable. If *ssthresh* is set to an arbitrarily high value, then the chances of a loss occurring before the slow start stage comes to an end are greatly increased. This is also expected due to reasonable loss probability values.

It follows that

$$E[L] = E[\alpha] + E[W] - 2 \tag{21}$$

Since the assumptions are the same as in section IV we have from (7)

$$E[\alpha] = \frac{1}{\left(1-(1-p)^2\right)} \tag{22}$$

Substituting (22) in (21) we have

$$E[L] = \frac{1}{\left(1-(1-p)^2\right)} + E[W] - 2 \tag{23}$$

At the beginning of the last round the window size would be W_i

where

$$W_i = 2^{\left(\frac{X_i}{b}-1\right)},\ i = 1,2,3\ldots \tag{24}$$

Note the assumption that $\frac{X_i}{b}$ is an integer.

Since in slow start , L_i packets are transmitted, the total number of packets sent would be

$$L_i = b * \left[2\left(\frac{X_i}{b}\right) - 1\right] + \beta_i \tag{25}$$

where β_i is the number of packets sent in the last round.

From (24) and (25) we have

$$L_i = b * (2W_i - 1) + \beta_i$$

The expected value of L is given by

$$E[L] = b * (2E[W] - 1) + E[\beta] \tag{26}$$

Substituting (21) in (26) we have

$$\frac{1}{\left(1-(1-p)^2\right)} + E[W] - 2 = b*(2E[W]-1) + E[\beta] \tag{27}$$

Since the number of packets sent in the last round have a uniform distribution between 0 and $2^{\left(\frac{X_i}{2}\right)} - 2$, or between 0 and $2W_i - 2$

$$E[\beta] = E[W] - 1 \tag{28}$$

hence substituting (28) in (27) we have

$$E[W] = \left[\frac{1}{2b} * \left(\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right) + \frac{1}{2}\right] \tag{29}$$

Substituting (29) in (23) we have

$$E[L] = \frac{1}{\left(1-(1-p)^2\right)} + \frac{1}{2b}\left[\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right] - \frac{3}{2} \tag{30}$$

B. Derivation of $E[Z^{SS}]$

Let r_j be the duration of the j^{th} round of the slow start stage under consideration. Hence the duration of slow start stage is

$$Z_i^{SS} = \sum_{j=1}^{X_i+1} r_j$$

Since r_j is considered to be a random variable, assumed to be independent of the size of the congestion window and hence independent of the round number *j*, it follows that

$$E\left[Z^{SS}\right] = (E[X]+1)\,E[r] \tag{31}$$

$E[r]$ is denoted by RTT, the average value of the round trip time for the connections in series under consideration, which in our case is the second connection of the split TCP connection. Now X_i is the round where a loss occurs for the first time in the slow start sequence under consideration. For the X_i^{th} round to start, it must be that packets in $X_i - 1$ rounds were transmitted successfully. In other words, that $2^{(X_i-2)}$ packets have been transmitted successfully.

So $P[X_i = k] = \left(2^{k-1} - 1\right)$ is the probability that packets have been transmitted successfully before the round where a loss occurs. So the expected value of X_i is

$$E[X] = \sum_{k=1}^{\infty} k\left((1-p)^2\right)^{\left(2^{(k-1)}-1\right)} \tag{32}$$

IX. Receiver's Window Limitation

A. Effects of Window Limitation

Let W_{max} denote the window size as advertised by the receiver. Hence in a period without loss the window size may grow up to W_{max} , but may not grow beyond this value. We make a simplifying assumption in our analysis. We denote by W_{u1} the unconstrained congestion control window size when the connection is in congestion avoidance and by W_{u2} the unconstrained congestion control window size while the connection is in slow start defined by equations (12) and (29).

We assume that when $E[W_{u1}] < W_{max}$ the receiver's advertised window limitation has negligible effect on the TCP throughput and hence $E[W] \approx E[W_{u1}]$, where $E[W]$ is the expected value of the congestion control window size when the connection is in congestion avoidance. But if $E[W_{u1}] \geq W_{max}$ then $E[W] \approx W_{max}$.

Similarly, when $E[W_{u2}] < W_{max}$ then $E[W] \approx E[W_{u2}]$, where $E[W]$ is the expected value of the window size when the connection is in slow start. But if $E[W_{u2}] \geq W_{max}$ then $E[W] \approx W_{max}$.

B. Congestion Avoidance

We first investigate the effect on throughput of the receiver's advertised window size while the connection is in congestion avoidance. Consider a sequence of triple duplicate periods. During any of the triple duplicate periods the window size grows linearly up to W_{max} for U_i rounds and then remains constant for V_i rounds at the end of which the current triple duplicate period comes to an end. Hence we have

$$W_{max} = \frac{W_{max}}{2} + \frac{U_i}{b} \tag{33}$$

From (33) it follows that

$$E[U] = \frac{b}{2} W_{max} \tag{34}$$

Counting the number of packets sent in the i^{th} TDP we have

$$Y_i = \frac{U_i}{b}\left(\frac{W_{max}}{2} + W_{max}\right) + V_i W_{max} \tag{35}$$

From (34) and (35) we have

$$E[Y] = \frac{3b}{8}W_{max}^2 + E[V]W_{max} \tag{36}$$

Substituting (11) in (36) we have

$$E[V] = \frac{(1-p)^2}{\left(1-(1-p)^2\right)W_{max}} + 1 - \frac{3b}{8}W_{max} \tag{37}$$

Since $X_i = U_i + V_i$ we have

$$E[X] = E[U] + E[V] \tag{38}$$

Substituting (34) and (37) in (38) we have

$$E[X] = \frac{b}{8}W_{max} + \frac{(1-p)^2}{\left(1-(1-p)^2\right)W_{max}} + 1 \tag{39}$$

Hence if $E[W_{u1}] \geq W_{max}$ then $E[W] \approx W_{max}$ then from (11) we have

$$E[Y] = \frac{(1-p)^2}{\left(1-(1-p)^2\right)} + W_{max} \tag{40}$$

and from (15) and (39)

$$E[A] = RTT\left(\frac{b}{8}W_{max} + \frac{(1-p)^2}{\left(1-(1-p)^2\right)W_{max}} + 2\right) \tag{41}$$

and

$$Q = Q\prime\,(W_{max}) \tag{42}$$

C. *Slow Start*

We next investigate the effect on throughput of the receiver's advertised window size while the connection is in slow start. During slow start the window size grows exponentially up to W_{max} for U_i rounds and then remains constant for V_i rounds at the end of which the current slow start sequence comes to an end. Counting the number of packets sent while this sequence of slow start lasts we have

$$L_i = \left(2^{\frac{U_i}{b}} - 1\right) + W_{max}V_i \tag{43}$$

Further, the sequence of slow start comes to an end at the end of $X_i = U_i + V_i$ rounds.

Using equation (34) we have

$$L_i = (2W_{max} - 1) + W_{max}V_i$$

It follows that

$$E[L] = (2W_{max} - 1) + W_{max}E[V] \tag{44}$$

Substituting equation (30) in (44) we have

$$E[V] = \frac{1}{\left(1-(1-p)^2\right)W_{max}} + \frac{1}{2bW_{max}}\left[\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right] - \frac{1}{2W_{max}} \tag{45}$$

From (33) we have

$$U_i = f(W_{max}) \tag{46}$$

where

$$f(W_{max}) = b\,(1 + log_2 W_{max})$$

From equation (46) it follows that

$$E[U] = E[f(W_{max})]$$

approximating this we may write

$$E[U] = f(W_{max})$$

or

$$E[U] = b\,(1 + log_2 W_{max}) \tag{47}$$

Since

$$E[X] = E[U] + E[V]$$

we have

$$E[X] = b\,(1 + log_2 W_{max}) + \frac{1}{\left(1-(1-p)^2\right)W_{max}} + \frac{1}{2bW_{max}}\left[\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right] - \frac{1}{2W_{max}} \tag{48}$$

Hence we have if $E[W_{u1}] \geq W_{max}$ then $E[W] \approx W_{max}$ then from (26) and (28) we have

$$E[L] = 2bE[W] + E[W] - 1 - b$$

or

$$E[L] = (W_{max}(2b+1)) - (b+1) \tag{49}$$

and

$$E[Z^{SS}] = (E[X]+1)\,RTT \tag{50}$$

where

$$E[X] = b\,(1 + log_2 W_{max}) + \frac{1}{\left(1-(1-p)^2\right) W_{max}} + \frac{1}{2bW_{max}} \left[\frac{(1-p)^2}{\left(1-(1-p)^2\right)}\right] - \frac{1}{2W_{max}}$$

X. Complete Characterization

The complete characterization of TCP throughput is given by

$$B = \frac{E[Y] + Q * (E[R] + E[L])}{E[A] + Q * (E\,[Z^{TO}] + E\,[Z^{SS}])} \tag{51}$$

defined by the following equations.

A. Case I

If $E[W_{u1}] < W_{max}$ and $E[W_{u2}] < W_{max}$ then equations (13), (16), (17), (18), (19). (20), (30), (31), (32) and (51) define the throughput.

B. Case II

If $E[W_{u1}] \geq W_{max}$ and $E[W_{u2}] \geq W_{max}$ then equations (17), (19), (20), (31), (40), (41), (42), (48), (49) and (51) define the throughput.

C. Case III

If $E[W_{u1}] \geq W_{max}$ and $E[W_{u2}] < W_{max}$ then equation (17), (19), (20), (30), (31), (32), (40), (41), (42) and (51) define the throughput.

D. Case IV

If $E[W_{u1}] < W_{max}$ and $E[W_{u2}] \geq W_{max}$ then equations (13), (16), (17), (18), (19). (20), (31), (48), (49) and (51) define the throughput.

XI. Simulation Results

We compared the expressions developed above against the results of simulations performed with ns-2 [7]. Ns-2 provides substantial support for simulation of TCP/IP networks, but has some limitations: TCP receivers do not provide a dynamic window advertisement; segment and ACK numbers are in units of packets, not bytes; and there is no SYN/FIN connection establishment/teardown. Further, no data is ever transferred, meaning that checksum computation time is not accounted for and it is impossible to model urgent data.

To run a TCP simulation one creates and configures a TCP sender *agent* then attaches it to a traffic generator. The TCP agent simulates the Reno form of TCP. Two applications are used as models for traffic generation: FTP and Telnet. FTP provides large, fixed size packets while Telnet varies its packet size randomly. The TCP agent supports various configurable variables such as the maximum packet size, receiver's (fixed) window size, initial slow start threshold value (*ssthresh*), etc.

One ns-2 limitation that most affected our work is that every node must be configured either as an agent or a sink, but not both. We would have preferred to configure an intermediate node that is both sink (for the first connection) and agent (for the second connection). However, ns-2 does not permit this, and the required changes are substantial. Accordingly, and because our analysis counts only the packets on the second connection, we simulated only the second TCP connection.

We simulated different durations varying from a few seconds to a few hours, using the FTP traffic generator. In our study, RTT is a parameter. However, in ns-2 RTT is a derived number, not a parameter. Two ns-2 parameters that influence RTT are bandwidth and link delay. In all simulations, these parameters were set to achieve the desired RTT. The random variable for generating errors was set to have a uniform distribution from 0 to 1. The packet, rather than the byte range, was the unit of error.

We compared the throughput predicted by our expressions against that generated by ns-2. We also compared the throughput predicted by our expression for a split connection against the throughput generated by ns-2 for a single connection.

A. Comparison Against Ns

We define Agreement as follows:

$$Agreement = \frac{Throughput\,as\,predicted\,by\,our\,expression}{Throughput\,generated\,by\,ns}$$

Since the idea is to compare the throughput as predicted by our expressions against those generated by ns-2, we will study the effect of varying various network parameters on Agreement.

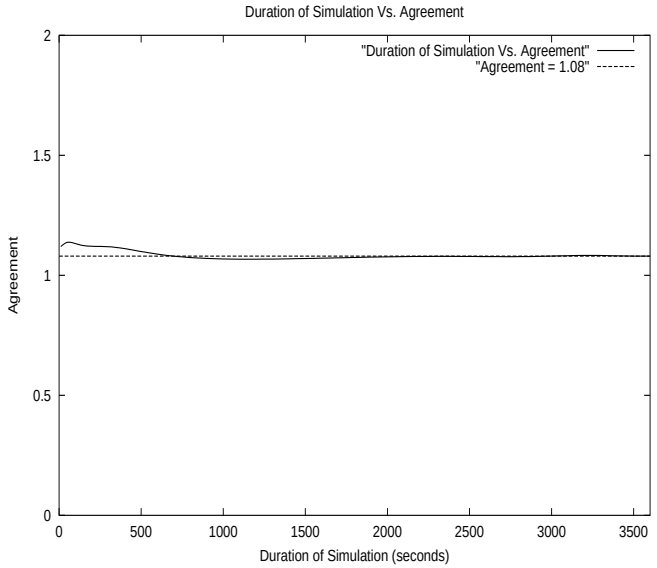

Fig. 1. Agreement vs. Simulation Duration
packet loss probability = 0.005
RTT = 0.005 seconds

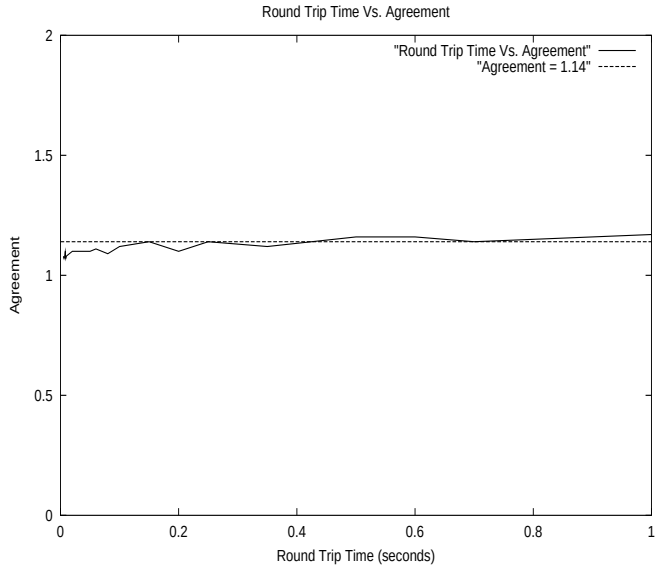

Fig. 2. Agreement vs. Round Trip Time
packet loss probability = 0.005

1) Agreement vs. Connection Duration: We carried out a series of simulations to determine the effect, if any, of the duration of the simulation on Agreement. Packet loss rate was varied in the interval $(0, 1)$ and round trip time was varied in the interval $(0, 1)$ seconds. The simulation duration varied from 10 seconds to one hour.

Figure 1 is one of the graphs plotted for reasonable values of packet loss rate and round trip time. The round trip time refers to the round trip time of the second connection. We determined "reasonable" values for these parameters by conducting simple informal experiments by pinging various sites on the Internet.

We found that there was practically no effect of the duration of simulation on Agreement. Therefore, duration of simulation can be eliminated as a parameter in further study; later simulations typically ran for an hour.

Agreement over the entire range of simulation durations was almost constant, at about 1.08. This means that, for the indicated parameters, the analytical expression and ns-2 agree to within 8%. For many different values of packet loss probability and round trip time, the curve was almost horizontal, only being displaced vertically above or below depending on the exact values of the parameters.

Packet loss probability of 0.005 and RTT of 0.005 seconds resulted in the best agreement of 1.08 and packet loss probabilities close to 1 and RTT value of 0.5 seconds resulted in the worst agreement of 1.9. As the RTT was gradually increased beyond 0.005 seconds to 0.5 seconds with the packet loss probability kept constant, the agreement moved from 1.08 to 1.25. Next, as the the packet loss probability was gradually increased beyond 0.005 to values close to 1, with the RTT kept constant at 0.005 seconds, the agreement moved sharply from 1.08 to 1.9. This indicated that the agreement was affected most by the packet loss probability and then by the RTT, though to a much lesser extent, and to almost negligible extent by the duration of simulation.

We next studied the effect of varying round trip time and packet loss probability on Agreement.

2) Agreement vs. Round Trip Time: For the particular value of packet loss probability (0.005) in Figure 2, Agreement is almost constant around 1.14 for a wide range of round trip times. When the value of the packet loss probability was varied in the interval $(0, 1)$, the curve still remained almost horizontal, in the vertical range $[0.9,\ 1.8]$.

A nearly horizontal Agreement curve is good, suggesting that our expression is valid for a wide range of round trip times. However, the fact that the horizontal line varies across $[0.9,\ 1.8]$ for different loss probabilities suggests that our expression and ns-2 do not model packet losses similarly, an effect suggested in the prior section and illustrated graphically in Figure 3.

3) Agreement vs. Packet Loss Probability: Figure 3 shows Agreement versus packet loss probability for a reasonable round trip time.

The shape of the graph is almost identical for all the plots drawn. We notice that for small values of the packet loss probability ($p < 0.0002$), the Agreement lies in the range $[1.16,\ 1.38]$. As the probability increases to 0.0005, the Agreement moves towards 1. It bottoms out at 0.9 for a packet loss probability of 0.0018. From then on, as the loss probability increases to 0.1, the Agreement gradually moves from 0.9 to 1.9, leveling off at 1.96 for a loss probability of 0.11. As the loss probability further increases all the way to 1, the Agreement slowly moves back to 1.

When the loss probability is small ($p < 0.0002$), the number

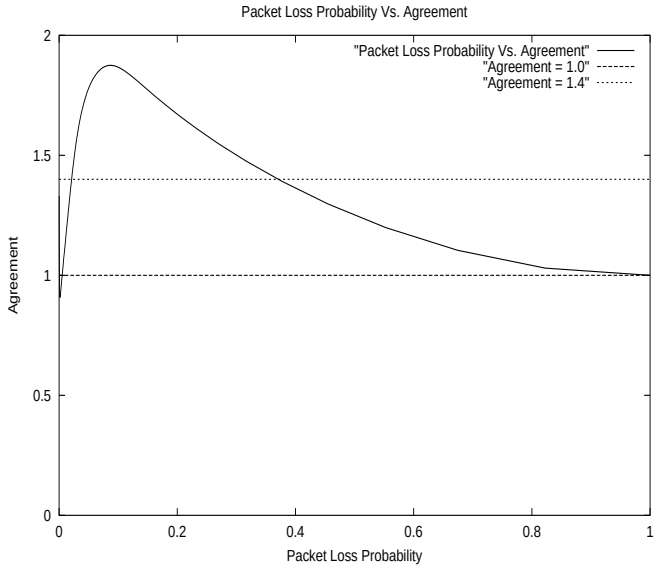

Fig. 3. Agreement vs. Loss Probability
round trip time = 0.05 seconds

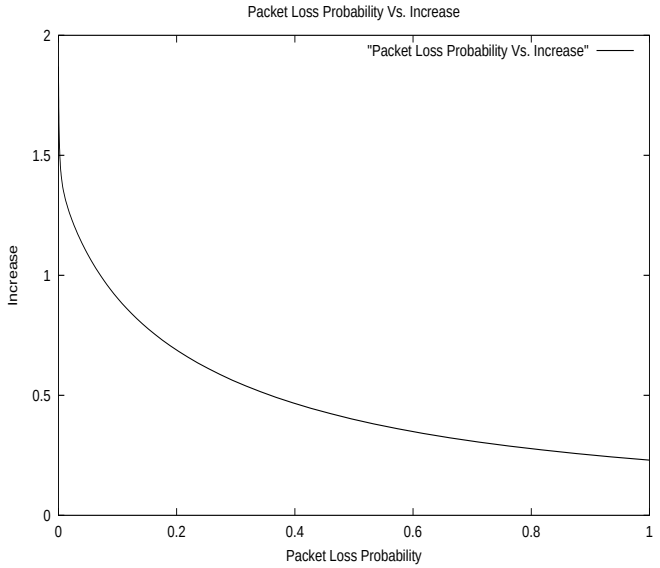

Fig. 4. Increase vs. Loss Probability
round trip time for the full connection = 0.5 seconds
round trip time for the split connection = 0.25 seconds

of packets transmitted is correspondingly large. Since the numbers being dealt with are very large a small shift might introduce a large error. Hence the Agreement is somewhat high. As the loss probability moves towards a more realistic (in terms of the Internet) value the Agreement is almost 1. In our analysis it has been found that expressions which do not model the slow start stage greatly over estimate the throughput. We have incorporated the slow start model in our equations. Hence the poor Agreement as the loss probability increases from 0.002 to 0.11 could be due to any or both of the following reasons

1) Our equations may not capture the entire essence of the evolution of the congestion control window as the connection moves from slow start to congestion avoidance and then oscillates between congestion avoidance, fast retransmit and slow start.
2) The values predicted by ns-2 do not exactly simulate the TCP connection as various congestion control algorithms come into play.

As the packet loss probability increases from 0.11 to almost 1 the Agreement starts getting better and tends to 1.0 for extremely high loss rates. This might be due to the fact that as the loss probability increases there are more timeouts than triple duplicate loss indications. Further, the number of packets transmitted is significantly reduced. Hence even a large shift may not make bring about a significant change in Agreement. Hence even though the graph might give the impression that the agreement is very good, there might be significant discrepancies.

B. Comparison with a Single TCP Connection

In this section we compare the throughput predicted by our expression for a split connection against the throughput generated by ns-2 for a single connection between the same endpoints. For the sake of simplicity we have assumed the following:

1) The processing time at the intermediate point is negligible and no packets are lost there.
2) The connection is split such that the round trip time of each of the two connections is one half the round trip time for the single connection. This has no bearing on the analysis – it has already been established that round trip time has little affect on Agreement.

We define $Increase$ to be the ratio of the throughput as predicted by our expression for a split connection to the throughput generated by ns for a single connection between endpoints. Increase was plotted against a wide range of packet loss probabilities. However we generated these plots for a whole range of round trip times.

Figure 4 shows the plot for a one set of round trip times. Plots for different RTTs are similar. We note from the graph that for reasonable loss probabilities of less than 0.05, the throughput predicted by the split connection is much greater than that of the single connection. However, as the packet loss probability increases above 0.05 the throughput of the split connection drops significantly compared to the single connection case.

XII. Conclusion and Future Work

We have corrected and extended an existing analytic model [1] to describe the performance of a *split* TCP connection, including slow start behavior. The equations specified in Section X provide a packet-based analytic characterization of a split TCP connection as a function of packet loss rate, average round trip time, and receiver's advertised window size. We computed the "Agreement" ratio between the values predicted by our expressions and those generated by ns-2 simulations as various network parameters were varied. We found that there is practically no effect of the duration of simulation on Agreement. Further, a

nearly horizontal Agreement curve suggests that our expression is valid for a wide range of round trip times. However, there is substantial divergence between our expression and ns-2 when packet loss is common.

We also compared the throughput as predicted by our expression for a split connection to the throughput generated by ns-2 for a single connection. We found that for reasonable loss probabilities (less than 0.05), the throughput predicted by the split connection is much greater than that of the single connection. However as the packet loss probability increases above 0.05, the throughput of the split connection drops significantly compared to the single connection case.

We noted in the simulations that, as the packet loss rate increased, the number of timeouts became significantly higher than triple duplicate loss indications. In these cases, the connection spends considerable time in slow start, thus making slow start modeling an important part of the analysis.

The idea that originally motivated this work – stringing together several transport connections, each terminated at an endpoint or intermediate node – suggests that perhaps many more than two connections might be arranged in series. As the number of connections in series is increased, the cumulative count of the packets in transit in all but the last connection will increase to become a sizable factor, large enough not to be neglected in analysis. Further analysis will need to be carried out for the general case of n TCP connections in series. Also, the assumption that a connection could be split such that all the intermediate links have the same RTT would no longer hold. Analyzing that in itself would be a significant research effort.

REFERENCES

[1] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. *Modeling TCP throughput: A Simple Model and its Empirical Validation,* in *Proc. ACM SIGCOMM*, pp. 303-314, September 1998.

[2] E. Altman, K. Avrachenkov and C. Barakat. *A Stochastic Model of TCP/IP with Stationary Random Losses,* in *Proc. ACM SIGCOMM*, pp. 231-242, August 2000.

[3] A. Bakre and B.R. Badrinath. *I-TCP: Indirect TCP for Mobile Hosts,* in *Proc. Fifteenth Intl. Conf. on Distributed Computing Systems*, May 1995.

[4] A. Bakre and B.R. Badrinath. *Implementation and Performance Evaluation of Indirect TCP,* IEEE Transactions on Computers, 46(3):260-278, March 1997.

[5] I. F. Akyildiz, G. Morabito and S. Palazzo. *Research Issues for Transport Protocols in Satellite IP Networks,* IEEE Personal Communications, 8(3):44-48, June 2001.

[6] H. Balakrishnan, V. Padmanabhan, S. Seshan and R.H. Katz. *A Comparison of Mechanisms for Improving TCP Performance over Wireless Links,* IEEE/ACM Trans. Networking, 5(6):756-769, December 1997.

[7] S. McCanne and S. Floyd. *ns-LBL Network Simulator*. Obtain via http://www.isi.edu/nsnam/ns/.

[8] P.L. Meyer. *Introductory Probability and Statistical Applications, Addision-Wesley*, 1970.

[9] P. Rodriguez, S. Sibal and O. Spatscheck. *TPOT: Translucent Proxying of TCP,* Computer Communication, 24(2):249-255, 2001.

[10] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. *The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm,* Computer Communication Review, 27(3):67-82, July 1997.

[11] A. Kumar. *Comparative Performance Analysis of Versions of TCP in a Local Network with a Lossy Link*, IEEE/ACM Trans. Networking, 6(4):485-498, August 1998.

[12] E. Altman, K. Avrachenkov and C. Barakat. *TCP in the Presence of Bursty Losses,* in *Proc. ACM SIGMETRICS*, pp. 124-133, June 2000.

[13] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby. *Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations*. IETF RFC 3135, June 2001.

[14] T. Ott, J.H.B. Kemperman, and M. Mathis. *The Stationary Behavior of Ideal TCP Congestion Avoidance*. Obtain via http://citeseer.nj.nec.com/ott96stationary.html.